

(2)

A COMPUTATION MODEL
OF ACQUISITION FOR
CHILDREN S ADDITION STRATEGIES

Technical Report AIP - 141

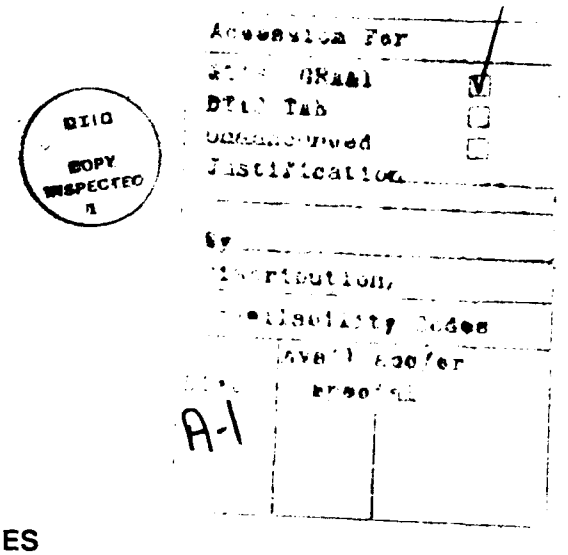
Randolph M. Jones and Kurt VanLehn

# The Artificial Intelligence and Psychology Project

Departments of
Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh






# A COMPUTATION MODEL OF ACQUISITION FOR CHILDREN S ADDITION STRATEGIES

Technical Report AIP - 141

Randolph M. Jones and Kurt VanLehn

Learning Research and Development Center
and Computer Science Department
University of Pittsburgh
Pittsburgh, PA

September 1991

This research was supported by the Computer Sciences Division, Office of Naval Research, under Contract Number N00014-86-K-0678. Reproduction in whole or in part is permitted for purposes of the United States Government. Approved for public release; distribution unlimited.

In L. Birnbaum and G. Collins (Eds.). Machine Leaning: Proceedings of the Eighth International Workshop. San Mateo, CA: Morgan Kaufman.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION / AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited. |
| 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>AIP-141 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Carnegie Mellon University<br>6b. OFFICE SYMBOL (*If applicable*) | 7a. NAME OF MONITORING ORGANIZATION<br>Computer Science Division<br>Office of Naval Research (Code 1133) |
| 6c. ADDRESS (*City, State, and ZIP Code*)<br>Department of Psychology<br>Pittsburgh, PA 15213 | 7b. ADDRESS (*City, State, and ZIP Code*)<br>800 North Quincy Street<br>Arlington, VA 22217-5000 |
| 8a. NAME OF FUNDING / SPONSORING ORGANIZATION<br>Same as monitoring organization<br>8b. OFFICE SYMBOL (*If applicable*) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>N00014-86-K-0678 |
| 8c. ADDRESS (*City, State, and ZIP Code*) | 10 SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. N/A; PROJECT NO. N/A; TASK NO. N/A; WORK UNIT ACCESSION NO N/A |

11. TITLE (*Include Security Classification*)
A computational model of acquisition for children's addition strategies

12. PERSONAL AUTHOR(S)
Randolph M. Jones and Kurt VanLehn

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (*Year, Month, Day*) | 15. PAGE COUNT |
|---|---|---|---|
| | FROM _____ TO _____ | August, 1991 | 4 |

16. SUPPLEMENTARY NOTATION In Birnbaum, L. and Collins, G. (Eds.) Machine Learning: Proceedings of the Eighth International Workshop. San Mateo, CA: Morgan Kaufman.

| 17. COSATI CODES | | | 18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | |
| | | | |
| | | | |

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

GIPS is a problem-solving system that models the strategy shifts of children learning to add. The system uses a generalized form of means-ends analysis as its reasoning algorithm, and it learns probabilistic selection and execution concepts for its operators. With this combination, GIPS models the SUM-to-MIN transition that children exhibit when learning to add(Siegler & Jenkins, 1989). The system generates the appropriate final strategy, as well as the intermediate strategies that Siegler and Jenkins observed.

| 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS | |

| 22a NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE (*Include Area Code*) | 22c. OFFICE SYMBOL |
|---|---|---|
| Alan L Meyrowitz | (202) 696-4302 | N00014 |

**DD FORM 1473,** 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Jones, R. and VanLehn, K. 1991. A computational model of acquisition for children's addition strategies. In Birnbaum, L., and Collins, G. (eds.) Machine Learning: Proceedings of the Eighth International Workshop. San Mateo, CA: Morgan Kaufmann. 65-69.

# A Computational Model of Acquisition for Children's Addition Strategies

Randolph M. Jones and Kurt VanLehn
Learning Research and Development Center
University of Pittsburgh
Pittsburgh, PA 15260

## Abstract

GIPS is a problem-solving system that models the strategy shifts of children learning to add. The system uses a generalized form of means-ends analysis as its reasoning algorithm, and it learns probabilistic *selection* and *execution* concepts for its operators. With this combination, GIPS models the "SUM-to-MIN" transition that children exhibit when learning to add (Siegler & Jenkins, 1989). The system generates the appropriate final strategy, as well as the intermediate strategies that Siegler and Jenkins observed.

## INTRODUCTION

Siegler and Jenkins (1989) have identified a number of distinct strategies that children exhibit when learning to add two numbers on their hands. This paper reports a model of the acquisition of these strategies in a computational problem-solving system. This model provides a testable theory of the cognitive mechanisms involved in learning to add. In addition, it has helped us identify some of the types of learning events and mechanisms that may be involved in general strategy acquisition.

The paper begins with a description of a computational problem solver called GIPS (General Inductive Problem Solver), which uses a generalized form of means-ends-analysis (MEA) (Jones, 1989). It's learning mechanism is based on Schlimmer's (1987; Schlimmer & Granger, 1986a, 1986b) STAGGER system, which uses probabilistic induction learn concepts from examples. The basic version of GIPS uses this algorithm to learn search-control knowledge (i.e., knowledge about when operators should be *selected*), similarly to other problem-solving systems that learn (e.g., SAGE, Langley, 1985; SOAR, Laird, Rosenbloom, & Newell, 1986; PRODIGY, Minton, 1988/1989). However, the enhanced version of GIPS also adjusts its representation of when operators can be *executed*. We believe that this ability is key to some of the strategy changes that people exhibit.

After describing the system, we present GIPS' account for the sequence of of addition strategies that Siegler and Jenkins found in children. GIPS successfully models the strategy shifts through a combination of it's general learning algorithm and simple changes to its *operator representations*. Toward the end of the paper, we provide a discussion of our results.

## THE GENERAL INDUCTIVE PROBLEM SOLVER

GIPS can be classified as a problem-solving system that learns from its experiences. However, it is notably different from other problem solvers in a number of ways. Primary among these are the use of a generalized form of means-ends analysis as the main planning algorithm, and a learning mechanism that is based on probabilistic reinforcement rather than a symbolic, analytical approach. In this section we describe the details of GIPS' planning and learning algorithms, together with its representation of operators and abstract problem descriptions.

### THE PLANNING ALGORITHM

In our research we have developed two versions of GIPS. To simplify the discussion, we will first describe the basic version of the system in detail. Later, we will discuss some of the additions we have made to develop a more complete model of strategy acquisition. The planning algorithm consists of the two functions TRANSFORM and APPLY. These functions behave similarly to systems that uses means-ends analysis. TRANSFORM attempts to satisfy what we call a TRANSFORM *goal* to change the current state into a state that satisfies some goal conditions. The current state and goal conditions are both represented as a set of relations over objects, where some of the objects may be variables. The function satisfies goals by first APPLYing an operator, and then recursively TRANSFORMing the resulting state. To APPLY an operator to the current state, the system must first TRANSFORM the current state into a state that satisfies the preconditions of the operator, and then EXECUTE the operator. We do not have the space to discuss the overall planning scheme in detail, but we will focus on the aspects that distinguish GIPS from other systems.

### SELECTION OF OPERATORS FROM MEMORY

As we have stated, this approach to problem solving is similar to a form of *means-ends analysis* (Ernst & Newell, 1969; Newell & Simon, 1972). However, GIPS uses a generalization of the standard approach, which is borrowed from the EUREKA system (Jones,

1989). Rather than always selecting operators whose actions mention the current goals, any operator in memory can theoretically be selected at any time. To determine which operators will actually be selected, GIPS borrows a probabilistic approach used by the PROSPECTOR expert system (Duda, Gaschnig, & Hart, 1979) and Schlimmer's (1987; Schlimmer & Granger, 1986a, 1986b) STAGGER system, which learns concept descriptions from examples. In the standard language for learning from examples, we say that each operator has associated with it a *concept* that predicts when it would be useful to select that operator. For the remainder of this paper, we will refer to this as the *selection concept* for an operator.

A TRANSFORM goal encountered during problem solving can be classified as either a *positive* or *negative instance* of a selection concept. Concepts and instances are represented as a set of *literals* (i.e., the relations that appear in TRANSFORM goals), which are matched in an attempt to classify the instances among the concepts. Finally, each literal in a concept has associated with it two values: *sufficiency* represents how much the presence of the literal predicts a positive instance of the concept, and *necessity* represents how much the absence of a literal predicts a negative instance of the concept.

When a new TRANSFORM goal is encountered, the literals of that goal are matched against the literals of each operator's selection concept in order to determine which of the concept's literals are present and which are absent. Once matching has occurred, the system calculates a prediction value that represents the odds that the current instance is a positive example of the concept. The formula used for prediction is

$$Odds(I \dashv C) = Odds(C) \prod_{f_j \mapsto I} S_j \prod_{f_j \not\mapsto I} N_j,$$

where $I \dashv C$ means that instance $I$ is a positive example of concept $C$, and $f_j \mapsto I$ means that literal $f_j$ of the concept matches a literal in $I$. Thus, the final prediction score represents the odds that instance $I$ is a positive example of concept $C$, and it consists of the product of the prior odds for the concept, the sufficiency scores of all the concept literals that are matched by the instance, and the necessity scores of all the literals that are not matched by the instance. If the odds are greater than one, it means that the instance is likely a positive example of the concept. In other words, given the current TRANSFORM goal, it is useful to attempt to APPLY the operator associated with this concept.

In STAGGER, matching is a trivial task because it uses a propositional representation. However, GIPS allows predicates, making matching more difficult. Each literal is a relation with a number of arguments, some of which may be variables. In addition, a relation can appear multiple times in an instance, each time with different argument values. Thus, there are generally multiple possible matches between an instance and a concept. Given a particular instance and concept, GIPS finds all the maximal partial matches of the literals in the instance to the literals in the concept. This means that there will be one prediction score for each *instantiation* of the instance to the concept.

### LEARNING IN GIPS

In the basic version of GIPS, learning involves changing the system's selection behavior. GIPS accomplishes this by first assigning credit and blame to the operators it has attempted to APPLY from each TRANSFORM goal. The TRANSFORM goal is classified as a positive instance of the selection concept for any operator that led to a solution from that goal. It is classified as a negative instance for operators that branch off of the solution path. The algorithm for storing examples in GIPS is based on the algorithm used in STAGGER. We have generalized Schlimmer's algorithm to handle a relational representation, but the current version of GIPS does not include STAGGER's constructive induction techniques. The learning algorithm depends on the statistical nature of the sufficiency and necessity scores for each literal. These scores are derived from conditional probabilities using the following formulas:

$$S_j = \frac{P(f_j \mapsto I | I \dashv C)}{P(f_j \mapsto I | I \not\dashv C)},$$

$$N_j = \frac{P(f_j \not\mapsto I | I \dashv C)}{P(f_j \not\mapsto I | I \not\dashv C)}.$$

The learning algorithm updates estimates of each of these conditional probabilities based on the presence of literals in the new instance and the classification of the instance as positive or negative. As the estimates change, so does GIPS' selection behavior on future problems. GIPS also augments its concept descriptions when the new instance contains literals that are not already present in the concept. It merely adds those literals to the concept description and, lacking any other knowledge about the importance of the literals, initializes the probability estimates for those literals to represent statistical independence.

## ADAPTING GIPS FOR STRATEGY ACQUISITION

The implementation of GIPS that we have described so far is capable of learning *search-control knowledge*, indicating when and in what order operators should be selected in new situations. In this version of GIPS, each operator has associated with it a probabilistic selection concept and a set of *preconditions* that specify when the operator can execute, among other things.

To enhance GIPS, we added to each operator a second probabilistic concept description that represents when the system thinks the operator should be able to *execute*. To execute an operator, the system no longer checks whether the preconditions are satisfied. Rather, it makes a probabilistic prediction based on the literals that are true in the current state, and it attempts

to execute an operator when the prediction value is greater than 1.

In order to learn these new *execution concepts*, the system cannot assign credit and blame itself without feedback from the outside world. Therefore, every time the system decides that an operator should execute, it asks the user to confirm its prediction. If the prediction is true, the system stores a positive example for this operator's execution concept and executes the operator. If the prediction is false, the system stores a negative example.

In addition to learning probabilistic execution concepts for operators, the enhanced version of GIPS handles two types of learning events that involve the preconditions of the operators. It is important for the system to be able to change the preconditions, because these literals are set up as subgoals when the operator cannot immediately execute. This has an impact on the selection of other operators as the system continues work on a problem. The first type of learning event occurs when the sufficiency value for a literal in the execution concept reaches a threshold. At this point, that literal is added as a precondition of the operator.

The second type of learning event occurs when the system successfully executes an operator, but not all of the preconditions are satisfied. In this case, the system removes the offending relations from the preconditions. This indicates that the system has found the preconditions to be an incorrect symbolic description of the execution concept. It is interesting to note that neither of these learning events are "impasse-driven," but they allow the system to gradually shift its representation of the domain it works in. These shifts manifest themselves as strategy changes when solving problems.

## REPRESENTATION OF THE ADDITION DOMAIN

The last system details concern GIPS' representation of the domain. GIPS describes the world as a set of relations between objects. In the addition domain, these objects and relations include the numbers that are part of the problem, the state of the problem solver's "hands" while it is adding, and the value of a counter that the problem solver keeps "in its head." The system also has a set of operators that simulate the solution of addition problems by novice problem solvers. Each operator includes a set of preconditions, add conditions, delete conditions, and possibly a set of constraints on variable bindings.

GIPS requires sixteen operators to represent the addition domain. There are two particular operators, which we refer to as the END-COUNT operators, that are involved in most of the strategy shifts. For future reference, the series of preconditions that the LEFT-END-COUNT operator acquires appears in Table 1. In addition to supplying the system with the operators, we initialized their selection concepts so the system generates the elementary adding strategy. We accomplished this by setting the literals of each operator's selection concept to be the preconditions and the goals that the operator could satisfy. Then, we initialized the conditional probabilities on these literals so that they would be selected in either a backward-chaining or forward-chaining fashion, depending on the role of the operator in the domain.

**Table 1.** A Series of Preconditions for LEFT-END-COUNT.

| |
|---|
| SUM strategy (a):<br>`Raising(Lefthand)`<br>`Counting(Lefthand)`<br>`Assigned(Lefthand,=Value)`<br>`Counter-value(=Value)` |
| SUM strategy (b):<br>`Raising(Lefthand)`<br>`Counting(Lefthand)`<br>`Assigned(Lefthand,=Value)`<br>`Counter-value(=Value)`<br>`Raised-fingers(Lefthand,=Value)` |
| SHORTCUT SUM strategy (c):<br>`Raising(Lefthand)`<br>`Counting(Lefthand)`<br>`Assigned(Lefthand,=Value)`<br>`Raised-fingers(Lefthand,=Value)` |
| FIRST strategy (d):<br>`Raising(Lefthand)`<br>`Counting(Lefthand)`<br>`Assigned(Lefthand,=Value)` |

To be more precise, some operators were initialized so that the literals representing goals were all highly sufficient for selection. Thus, they would be selected any time one of the system's current goals matched an operator action. For the forward-chaining operators, all of the literals representing the operator preconditions were initialized as highly necessary. These operators would not be selected unless all of the preconditions could be matched by the current state. The combination of forward-chaining and backward-chaining operators allows the system to generate more complex (and more psychologically plausible) reasoning behavior than would be allowed by a strictly forward-chaining or means-ends-analysis system.

## STRATEGY ACQUISITION IN THE ADDITION DOMAIN

This section presents GIPS' behavior through a series of different strategies for adding numbers. These strategy shifts arise from the learning algorithm incorporated into the system, and they correspond to actual strategies that children acquire when learning the task (Siegler & Jenkins, 1989).

## THE SUM STRATEGY

GIPS' initial strategy for addition corresponds to the SUM strategy found in children. In this strategy, the problem solver attempts to add two numbers by first setting up the proper number of fingers on each hand (*representing* the addends) and then counting up the fingers. The first thing the system does is assign an addend to each hand. For example, for the problem of $3 + 2$, the system might assign the number 2 to the left hand (the first hand) and the number 3 to the right hand. However, in this strategy the order of the addends does not make a difference, so it could just as easily have switched them.

In continuing the problem, the system uses a single counter together with its hands to generate an answer. The system raises its fingers and counts them one at a time until the counter value is equal to the value of the appropriate addend. This indicates that an END-COUNT operator should execute. We feel that the counter plays this role because, after representing one addend, children reset their count to zero in order to represent the second. If the counter were not being used to stop the count, it would not have to be reset between hands.

As the system solves new addition problems, it updates the execution concepts for the END-COUNT operators. It soon notices a number of relations that are always true when these operators execute. The most important of these is that the number of raised fingers is equal to the counter value. This and other relations get added into the preconditions for the END-COUNT operators (see Table 1(b)). This action alone does not change the system's outward behavior, but it proves important for subsequent strategies.

## THE SHORTCUT SUM STRATEGY

After some time, the new literals in the system's execution concepts for LEFT-END-COUNT and RIGHT-END-COUNT become so strong that it attempts to execute the operators earlier than usual. At this point, GIPS thinks that the operator should execute when the number of fingers raised on a hand is equal to the goal value, even though the system has not yet incremented its count for the last finger. It turns out that the system can successfully solve the addition problem even if it executes this operator prematurely, so it deletes the condition that the current counter value must be equal to the goal value in the preconditions of the END-COUNT operators (see Table 1(c)).

This change has a direct effect on GIPS' behavior. It continues to increment its counter every time it raises a finger, but it no longer resets the counter when it is done representing an addend. This is because the value of the counter is no longer used as the termination criterion to stop counting a hand. Because the counter is not reset between hands, there is no need to go back and count up all the fingers on both hands after the addends have been represented. This behavior corresponds to the SHORTCUT SUM strategy used by children.

## THE "FIRST" STRATEGY

The next strategy shift occurs similarly. As GIPS attempts to execute the END-COUNT operators at various times with feedback from the user, it develops a "good" concept for when the END-COUNT operators are executable. One important part of this concept is that the goal value for counting fingers on a hand is always equal to one of the addends when LEFT-END-COUNT executes.

Eventually, the system attempts to fire the LEFT-END-COUNT operator without having raised any fingers at all. When it succeeds, it deletes the precondition that the number of fingers raised on the hand be equal to the goal value (see Table 1(d)). The system has learned that it can simply start counting from the goal value for the left hand rather than starting from zero. Note that there is no way that the system could have jumped to this strategy from the initial strategy. This indicates that a noise-tolerant, reinforcement approach is appropriate to account for this series of strategies. GIPS also attempts to execute the RIGHT-END-COUNT operator early, but this leads to failure. Thus, the system begins to exhibit the FIRST strategy, in which the first number (or lefthand number) is simply announced and used to continue counting the second number as it did in the SHORTCUT SUM strategy.

## THE MIN STRATEGY

The final strategy that GIPS generates is the MIN strategy. MIN is similar to the FIRST strategy, except that the system learns that it should not assign the addends arbitrarily to its hands. Rather, it starts with the larger addend, and continues counting with the smaller, resulting in less work. In GIPS, the knowledge required to generate this strategy can be learned during the SHORTCUT SUM or FIRST strategies. In both of these strategies, when the problem solver is representing an addend on the right (or second) hand, the counter value is not equal to the number of fingers that are raised.

We hypothesize that a student may sometimes get mixed up or lose count and fail to solve the problem because he "loses his place" when representing the right-hand addend. This type of interference would have more chance of occurring for larger numbers. Thus, the solver would learn to prefer counting the smaller number on the right hand, because it leads to fewer failures of this type. This type of failure would not occur with the left hand, because the number of raised fingers in the SHORTCUT SUM strategy is always equal to the value of the counter for that hand.

We simulated this hypothesis in GIPS by causing it to fail sometimes during the SHORTCUT SUM strategy when it decided to count the larger addend on its

right hand. This caused the system to update the selection concept for the operator that assigns numbers to hands, so that it would prefer to assign the smaller addend to the right hand. With this experience, the system developed the MIN strategy rather than the FIRST strategy.

## DISCUSSION

We have introduced a new learning problem-solving system, called GIPS. Although most systems in this area use an "explanation-based" approach, GIPS incorporates a probabilistic, reinforcement learning algorithm. This learning algorithm has been demonstrated to account for some low-level learning behaviors (Rescorla, 1968; Schlimmer, 1986), and our success with GIPS indicates that it is suitable to model learning at higher levels required for problem solving.

GIPS uses this probabilistic approach to learn selection and execution concepts for its operators. Subtle changes in these concepts result in sometimes dramatic shifts in strategy while solving problems. In the domain of addition, we have demonstrated that the strategies acquired by GIPS match strategies acquired by children. This argues that the learning and reasoning mechanisms incorporated into GIPS correspond (at some level) to cognitive mechanisms in the children. One other computational system models the SUM-to-MIN transition (Neches, 1987), but GIPS appears to provide a much better account of the psychological data found by Siegler and Jenkins (1989).

This study concentrated on providing a qualitative account for a set of general psychological behaviors. Our next step is to use GIPS to model human behavior at a smaller grain size. Even within general strategies, there is a large amount of room for individual differences among subjects, and we feel that the mechanisms in GIPS are ideal for capturing these sometimes subtle distinctions.

### Acknowledgments

This research benefited from discussions with Bob Siegler and Jeff Schlimmer. It was supported in part by contract N00014-88-K-0080 from the Office of Naval Research, Cognitive Sciences Division, and a postdoctoral training grant from the Department of Health and Human Services.

### References

Duda, R., Gaschnig, J., & Hart, P. (1979). Model design in the Prospector consultant system for mineral exploration. In D. Michie (Ed.), *Expert systems in the micro electronic age*, Edinburgh: Edinburgh University Press.

Ernst, G., & Newell, A. (1969). *GPS: A case study in generality and problem solving.* New York: Academic Press.

Jones, R. (1989). *A model of retrieval in problem solving.* Doctoral dissertation, University of California, Irvine.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning, 1*, 11–46.

Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science, 9*, 217–260.

Minton, S. (1989). Learning effective search control knowledge: An explanation-based approach (Doctoral dissertation, Carnegie Mellon University, 1988). *Dissertation Abstracts International, 49*, 4906B–4907B.

Neches, R. (1987). Learning through incremental refinement of procedures. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development.* Cambridge, MA: MIT Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice Hall.

Rescorla, R. A. (1968). Probability of shock in the presence and absence of CS in fear conditioning. *Journal of Comparative and Physiological Psychology, 66*, 1–5.

Schlimmer, J. C. (1986). *A note on correlational measures* (Tech. Rep. No. 86-13). Irvine: University of California, Department of Information and Computer Science.

Schlimmer, J. C. (1987). Incremental adjustment of representations for learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 79–90). Irvine, CA: Morgan Kaufmann.

Schlimmer, J. C., & Granger, R. H., Jr. (1986a). Beyond incremental processing: Tracking concept drift. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 502–507). Philadelphia: Morgan Kaufmann.

Schlimmer, J. C., & Granger, R. H., Jr. (1986b). Incremental learning from noisy data. *Machine Learning, 1*, 317–354.

Siegler, R. S., & Jenkins, E. (1989). *How children discover new strategies.* Hillsdale, NJ: Lawrence Erlbaum.